# Git Topics

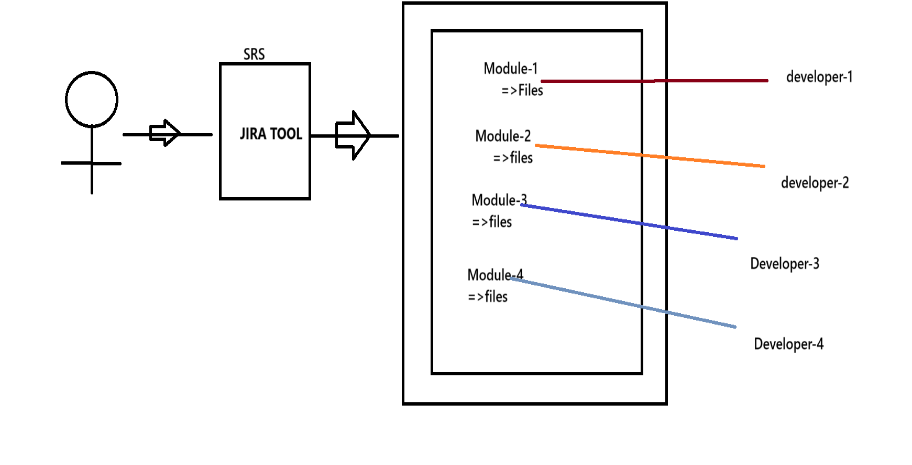
**1.Introduction**

🡺Git is a popular Version Control System(VCS)

🡺It was created by Linus Torvalds in 2005 and it is maintained by Junio Hamano

Git is used for

1. Tracking code changes
2. Tracking who made the changes like history of Files
3. Coding Collabarations



Keeping Track of the changes made to the files by the developer as per the changes made by the client requirement would be difficult in developer machine. To resolve this problem we need to use “version Control System”.

What is Version Control System(VCS) and types of VCS?

It is a system that records changes made to the file or set of files over the time, so that we can recall the specific version later. i.e. for every source code changes in a file a new version will be created.

Types of Version Control Software(VCS)

There are 3 types of VCS

**a. Local Version Control System**

**b. Centralized Version Control System**

**c. Distributed Version Control System**

# **a. Local Version Control System**

**🡺**It is used to maintain the file version and retrieve the file based on the specific version.

**Drawbacks**

1. It is easy to forget in which drive you are in and accidentally write the data to the worn file or copy from other files.
2. If the hard disk is corrupted there would a possible loss of secured data.
3. By mistake we can delete few file also.

To overcome the drawback of Local VersionControl System we have “Centralised version Control System”.

**Centralised Version Computer**

**🡺**Developers can collaborate the code in one repository and do the change.

e.g. of Centralised Version software :- SVN,Subversion,perforce,…..

🡺Centralised Version server will have single server that contains all the version files.

🡺For many years this has been the standard version control system.

🡺More no. of developers would connect to CVS to check out the files.

Notes:-

🡺Checkout🡺taking the code from repository to local machine.

Push🡺sending the code from local machine to repository(CVS)

Advantage

1. Every One know to certain degree what everyone else on the project is doing.

2. Administrator will have full control over whoch can do what and it is easier to manage.

Drawback

-------------

1. Single point of Failure(SPF) would represent the Centralized system.

2. If the server goes down due to network traffic, during that hour nobody can collaborate at all or save changes to the server.

3. If the hard disk of the centralized system gets corrupted and proper backup haven't been taken then there is every possibility of loss of data.

**Note**: In LVCS ad in CVCS getting up the complete history of changes is not possible.

It is possible to get only the latest version, but not the entire history.

eg: SVN

push will not happen w.r.t version rather push will happen only with

the latest change.

Version history

file -> 1.0V

file-> 1.1V

file-> 1.2V

file-> 1.3V

Distrubuted Version Control System

=============================

Eg: Git, Mercurial, Darcs, Baazar, etc,....

=> Developers will not only get the latest version but also the compile history of the files

=> Push will not only happen with latest snapshot of the files rather they will push the old files also.

=>If the main sever goes off, still there is a local repository which would have maintained the copy of the repository

where the entire code is available(history of versions).

=> If the remote repository is down, then developers can do changes in the local repository and when the main repository is up the code can be pushed to remote repository from local repository.

Git installation

There are 2 types of Git software

1. Git Server

2. Git Client

**GitServer**

-> It is a repository

-> It is the largest host of source code in the world.

-> It is used to store/manage the source code of the project

-> Some of the Git server tools are : Github,BitBucket,GitLab,....... GitServer(Github,gitlab)

**How to connect to the Gihub?**

Cards(Source code) Payment(Source co)

Offers(Source code)

Loans(Source code

We need

URL🡪http://repo.citibank:9999

Username🡪

Password🡪

Git URl will be save for all the developers but username and password

Will be different for all the developers.

Note:- When we join a company team leader or manager will share the URL

Username, password. Every developers will connect to git Server and get

Source code from the git server and do the changes locally and then move

the code from the local repository to the main repository.

Where should we provide url, username and password?

To type these details we need git client.

GitClient

Installation of git software

1. Download a git software from the following link

<https://git-scm.com/download/win>

It is a tool which is used to connect to our gitserver.

if we install git client(git s/w) we get the following tools for free

a. git bash => linux commands are required

b. git gui => Graphical user interface where all the actions will be done through clicks

c. git cmd => command line tools where developer should provide url,username and password

**Note:**

Git client is a .exe file which can installed with just few clicks.

git -> client tool where the client should provide url, username and password

git hub-> server software where repository/projects will be maintained.

**Git Architecture**

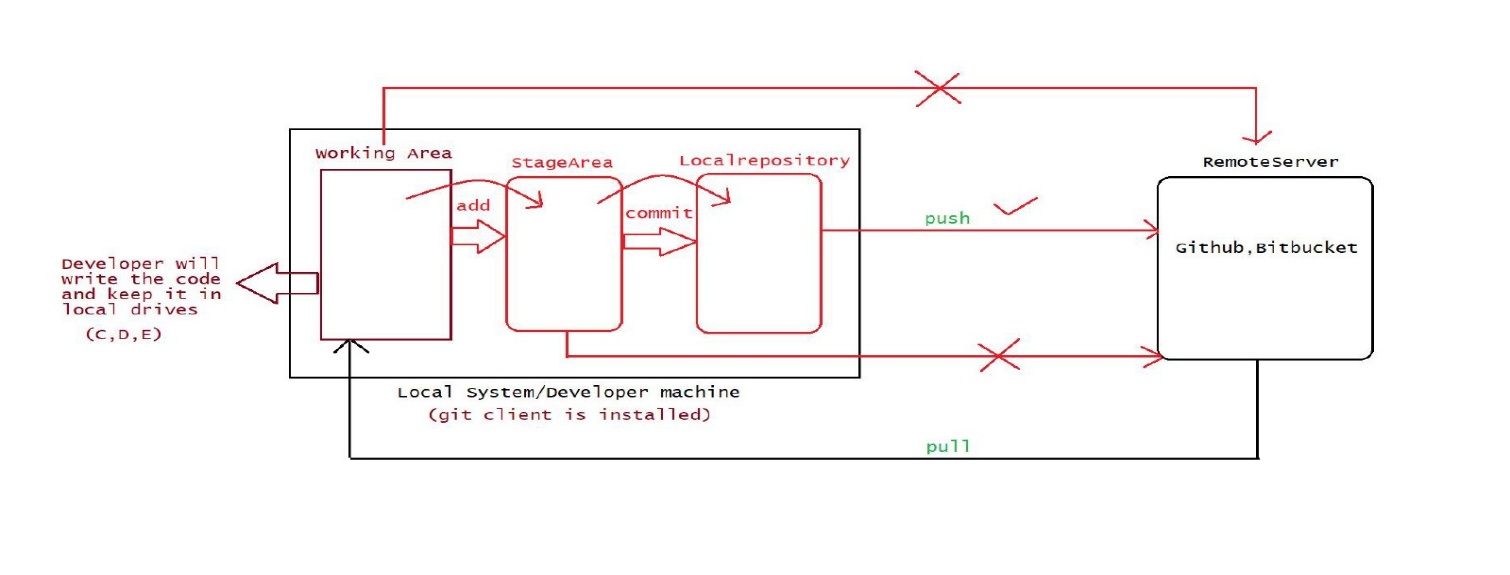
---------------------

There are 3 regions

a. workplace => It is a place where developers maintain there source code

b. stage area => Once the code is ready, then it will be added to stage area(indication to git software)

c. local repository=> Once the code is in stage area, we commit it to the local repository with some standard message, From local repository we "push" to main repository by providing url, username and password.



**Git Commands:-**All the commands are case sesitive

1.Git version 9.Git config 17.git Checkout

2.Git help 10.Git Commit 18.Git Stash

3. Git init 11.Git log

4.Git Clone 12.Git Show

5. Git add 13.Git Push

6. Git Status 14.Git CLone

7. Git RM 15. Git pull

8.Git restore 16.Git branch

**1.git version**

This Command is used to check the version of the git.

Gaurav Kumar Gupta@MSI MINGW64 ~

$ git version

git version 2.49.0.windows.1

**2.git help**

If we want to see the list of commands then we can use get help

Gaurav Kumar Gupta@MSI MINGW64 ~

$ git help

Note:- This command is useful to get the documentation of any Command

e.g. git command<Command-name>

**3.git config**

It is used when the git software is used for the first time

This command will set the developer identity like name, emaild..

This configuration information will be used by git software for every push operation encountered.

🡪**git config --list** // This command is used to provide list of configuration or it show from which email id it is registered

Gaurav Kumar Gupta@MSI MINGW64 ~

$ git config list

diff.astextplain.textconv=astextplain

filter.lfs.clean=git-lfs clean -- %f

filter.lfs.smudge=git-lfs smudge -- %f

filter.lfs.process=git-lfs filter-process

filter.lfs.required=true

http.sslbackend=schannel

core.autocrlf=true

core.fscache=true

core.symlinks=false

pull.rebase=false

credential.helper=manager

credential.https://dev.azure.com.usehttppath=true

init.defaultbranch=master

**user.name=Gaurav**

[**user.email=gupta.gaurav9955@gmail.com**](mailto:user.email=gupta.gaurav9955@gmail.com)

**🡪git config --list --show-origin //**This command will show where your configuration file is stored in your computer with **.gitConfig** name.

Gaurav Kumar Gupta@MSI MINGW64 ~

$ git config --list --show-origin

file:C:/Program Files/Git/etc/gitconfig diff.astextplain.textconv=astextplain

file:C:/Program Files/Git/etc/gitconfig filter.lfs.clean=git-lfs clean -- %f

file:C:/Program Files/Git/etc/gitconfig filter.lfs.smudge=git-lfs smudge -- %f

file:C:/Program Files/Git/etc/gitconfig filter.lfs.process=git-lfs filter-process

file:C:/Program Files/Git/etc/gitconfig filter.lfs.required=true

file:C:/Program Files/Git/etc/gitconfig http.sslbackend=schannel

file:C:/Program Files/Git/etc/gitconfig core.autocrlf=true

file:C:/Program Files/Git/etc/gitconfig core.fscache=true

file:C:/Program Files/Git/etc/gitconfig core.symlinks=false

file:C:/Program Files/Git/etc/gitconfig pull.rebase=false

file:C:/Program Files/Git/etc/gitconfig credential.helper=manager

file:C:/Program Files/Git/etc/gitconfig credential.https://dev.azure.com.usehttppath=true

file:C:/Program Files/Git/etc/gitconfig init.defaultbranch=master

**file:C:/Users/gauta/.gitconfig** user.name=Gaurav

**file:C:/Users/gauta/.gitconfig** [user.email=gupta.gaurav9955@gmail.com](mailto:user.email=gupta.gaurav9955@gmail.com)

**To Set the username and email**

**>git config --global user.name “Gaurav Gupta”**

**>git config --global user.email** [**gupta.gaurav9955@gmail.com**](mailto:gupta.gaurav9955@gmail.com)

**global🡪** It indicates the user can work with git commands from different drive of the computer.

**Important Operation Associated with git**

**3.git init 🡪** Transform the current directory into git directory

**🡺**Normally a folder will be created in the developers works place and inside the folder the source code would be place.

🡺Normally this is the first command which we execute to set up the git for operation like clone, push, pull

Let’s see how we are going to do it , we will first create the folder inside F drive with name **gitsession** and inside this folder we will create another folder with name **Workspace-1** and then we create that folder git folder.

**>cd F:**

**>mkdir gitsession**

**>cd gitsession/**

**>mkdir Workspace-1**

**>cd Workspace-1/**

**>git init**

🡺This Command internally created one folder called .git

🡺.git is used by git software to identify the folders which should participate in pushing to “local” and “remote” repository.

**4. git status**

This command is used to check the status of the working directory.

Gaurav Kumar Gupta@MSI MINGW64 /f/gitsession/Workspace-1 (master)

$ git status

On branch master

No commits yet

Untracked files:

(use "git add <file>..." to include in what will be committed)

Java Notes.docx

**git status normally give outputs in the following ways:-**

**a.Untracked files(red Color)->** It means the files are present still in working area and these files can’t be commit to “local repository” nor to “remote repository”

**b.Tracked Files(green color)->**It means the files are moved from working area to stage area so these files can be commited to “local repository” and remote repository.

**c Modified Files(Red color)->**It means the files are present still in working area and these file can be staged or it can be also restored back to the normal phase.

**5. git add or (git add .-->To add multiple file at once):-**

To transfer the file from the working directory to the staged area so these file can commited to local or remote repository we use this command

Like :-

Gaurav Kumar Gupta@MSI MINGW64 /f/gitsession/Workspace-1 (master)

$ git add 'Java Notes.docx'

**6.git rm –cached**

To transfer the file from staged area to working area so as to be untracked we use this command:-

Like

Gaurav Kumar Gupta@MSI MINGW64 /f/gitsession/Workspace-1 (master)

$ git rm --cached 'Java Notes.docx'

rm 'Java Notes.docx'

Now suppose you make an change to the file which is being transferred to the stage area and then you types a git status it will bounce back with the status as following:-

**Changes not staged for commit:**

**(use "git add <file>..." to update what will be committed)**

**(use "git restore <file>..." to discard changes in working directory)**

**modified: Java Notes.docx**

In this message we can see git restore<file> which is speaking about the restoration of file at the previous stage.

**7.git restore<filename>**

This command is used to restore the untracked file into the previous version of stage area.

**8.git commit**

The files which are ready to commit should be in stage area, to perform commit operation we use the following command.

Syntax:-

git commit –m <some messages>

ex

Gaurav Kumar Gupta@MSI MINGW64 /f/gitsession/Workspace-1 (master)

$ git commit -m 'Java\_notes'

[master (root-commit) 6b2815a] Java\_notes

1 file changed, 0 insertions(+), 0 deletions(-)

create mode 100644 Java Notes.docx

Now suppose if want to commit only one file then the syntax will be:-

Git commit –m ‘message’ <filename>

**Steps followed to create a remote repository and push it to remote repository**

**1.open the github.com by providing the credentials**

**2.Create the remote repository and enter some name(repository name) and click on remote repository.**

**3.To perform push operation we need to use the following command**

git branch -M main

git remote add origin https://github.com/GauravGupta9955/sample-project.git

git push -u origin main

